DIP 6th Lecture

October 3, 2023

1 Lecture 5: Digital Image Processing

Point Pixel Operations

1.0.1 Objectives

- To understand the concept of point pixel operations

1.0.2 Importing the required libraries

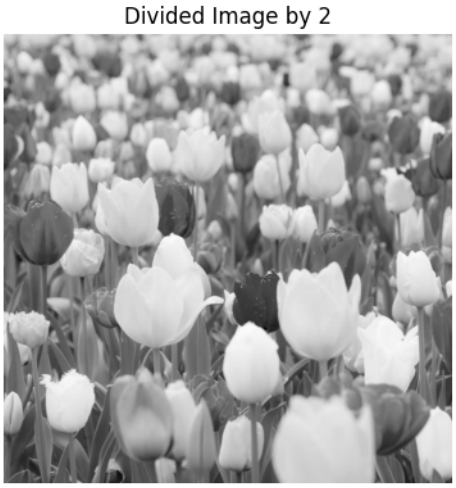
[ ]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import math
```

1.0.3 Reading the Image

Reading the image using the imread() function of the cv2 library and storing it in the variable image. Convert the image to grayscale using the cvtColor() function of the cv2 library.

[ ]:
```
image = cv2.imread("image1.jpg")
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

plt.figure(figsize=(12, 10))

# Plot the color image
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for␣
 ↪proper display
plt.title("Grayscale Image")
plt.axis("off")

# Plot the gray image
plt.subplot(1, 2, 2)
plt.imshow(image_gray, cmap="gray")
plt.title("Divided Image by 2")
plt.axis("off")
```

```
plt.show()
```



1.0.4 Binerization using Thresholding

Write a method which takes a gray image and return a binerized image using thresholding method. method should take image as perameter and return binerized image.

[ ]:
```
def binarize(image):
    image_bi = np.array(image)

    # Binarize the image
    for i in range(image_bi.shape[0]):
        for j in range(image_bi.shape[1]):
            if image_bi[i][j] >= 128:
                image_bi[i][j] = 1
            else:
                image_bi[i][j] = 0

    return image_bi
```

[ ]:
```
# Binarize the image
image_bi = binarize(image_gray)

plt.figure(figsize=(12, 10))

# Plot the color image
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for␣
 ↪proper display
```

```
plt.axis("off")

# Plot the binary image with the "binary" colormap
plt.subplot(1, 2, 2)
plt.imshow(image_bi, cmap="binary")
plt.axis("off")

plt.show()
```

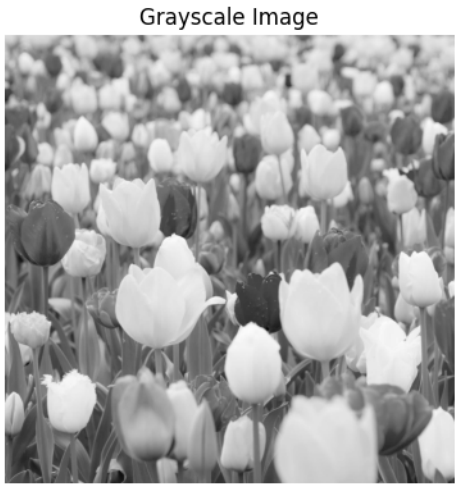
1.0.5 Multply Image by a Constant

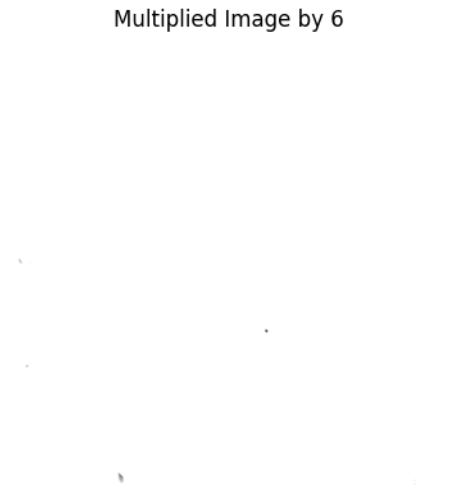
Write a method which takes a gray image and a constant as perameter and return a new image which is the multiplication of the image and the constant.

[ ]:
```
def multiply(image, contstant):
    image_mul = np.array(image)

    # Multiply the image with the constant
    for i in range(image_mul.shape[0]):
        for j in range(image_mul.shape[1]):
            if image_mul[i][j] * contstant > 255:
                image_mul[i][j] = 255
            else:
                image_mul[i][j] = image_mul[i][j] * contstant

    return image_mul
```

[ ]:
```
# Multiply the image with the constant
image_x2 = multiply(image_gray, 2)
image_x4 = multiply(image_gray, 4)
```

```
image_x6 = multiply(image_gray, 6)

plt.figure(figsize=(12, 10))

# Plot the grayscale image
plt.subplot(2, 2, 1)
plt.imshow(image_gray, cmap="gray")
plt.title("Grayscale Image")
plt.axis("off")

# Plot the multiplied image
plt.subplot(2, 2, 2)
plt.imshow(image_x2, cmap="gray")
plt.title("Multiplied Image by 2")
plt.axis("off")

plt.subplot(2, 2, 3)
plt.imshow(image_x4, cmap="gray")
plt.title("Multiplied Image by 4")
plt.axis("off")

plt.subplot(2, 2, 4)
plt.imshow(image_x6, cmap="gray")
plt.title("Multiplied Image by 6")
plt.axis("off")

plt.show()
```



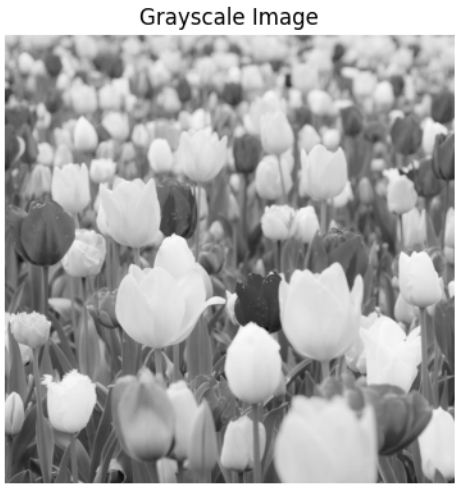


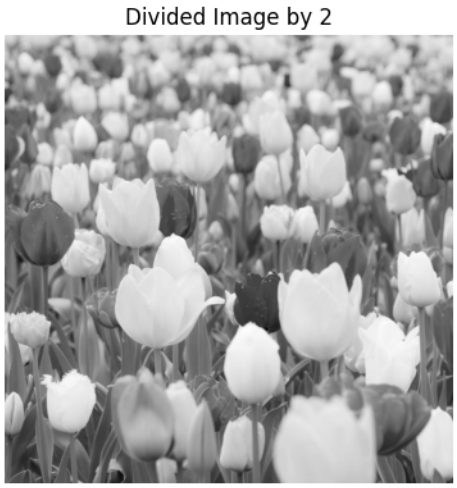
1.0.6 Divide Image by a Constant

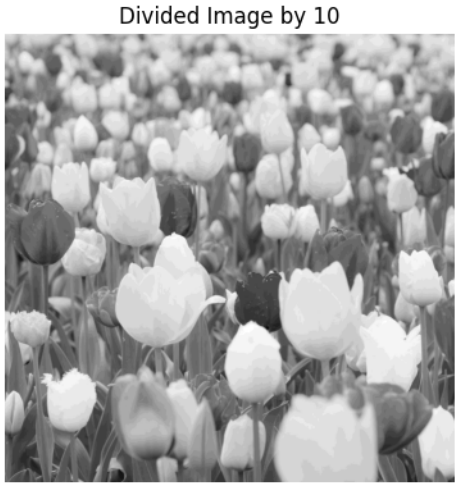
Write a method which takes a gray image and a constant as perameter and return a new image which is the division of the image and the constant.

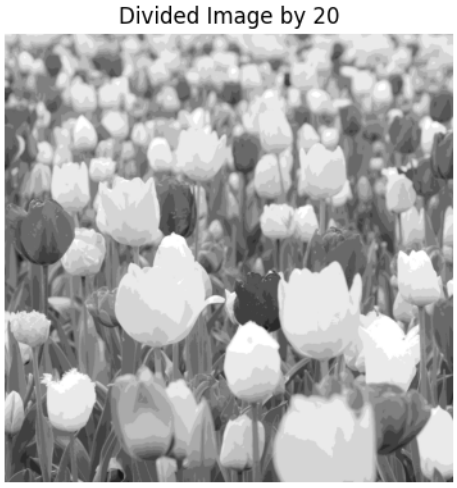
[ ]:
```python
def divide(image, constant):
    image_div = np.array(image)

    # Divide the image with the constant
    for i in range(image_div.shape[0]):
        for j in range(image_div.shape[1]):
            image_div[i][j] = math.floor(image_div[i][j] / constant)

    return image_div
```

[ ]:
```
# Divide the image with the constant
image_div2 = divide(image_gray, 2)
image_div10 = divide(image_gray, 10)
image_div20 = divide(image_gray, 20)

plt.figure(figsize=(12, 10))

# Plot the grayscale image
plt.subplot(2, 2, 1)
plt.imshow(image_gray, cmap="gray")
plt.title("Grayscale Image")
plt.axis("off")

# Plot the divided image
plt.subplot(2, 2, 2)
plt.imshow(image_div2, cmap="gray")
plt.title("Divided Image by 2")
plt.axis("off")

plt.subplot(2, 2, 3)
plt.imshow(image_div10, cmap="gray")
plt.title("Divided Image by 10")
plt.axis("off")

plt.subplot(2, 2, 4)
plt.imshow(image_div20, cmap="gray")
plt.title("Divided Image by 20")
plt.axis("off")

plt.show()
```







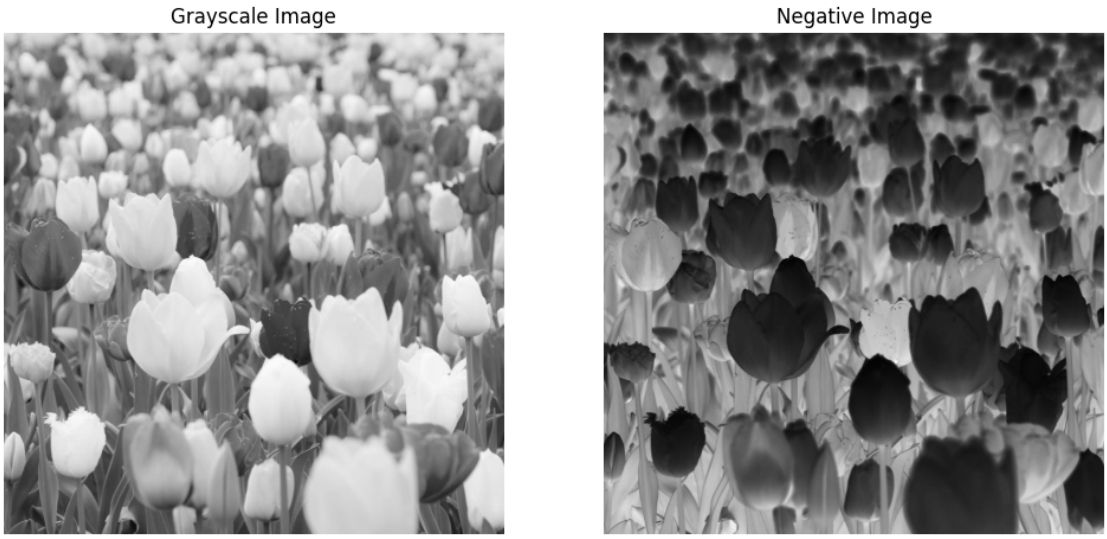


1.0.7 Negative Image

Write a method which takes a gray image as perameter and return a new image which is the negative of the image.

[ ]:
```
def negative(image):
    image_neg = np.array(image)

    # Negative the image
    for i in range(image_neg.shape[0]):
        for j in range(image_neg.shape[1]):
            image_neg[i][j] = 255 - image_neg[i][j]

    return image_neg
```

[ ]:
```
# Negative the image
image_neg = negative(image_gray)

plt.figure(figsize=(12, 10))

# Plot the grayscale image
plt.subplot(1, 2, 1)
plt.imshow(image_gray, cmap="gray")
plt.title("Grayscale Image")
plt.axis("off")

# Plot the negative image
plt.subplot(1, 2, 2)
plt.imshow(image_neg, cmap="gray")
plt.title("Negative Image")
plt.axis("off")

plt.show()
```



1.1 Conclusion

In this lab, we have learned how to perform point pixel operations on an image.